QBUS6850

Lecture 7

Advanced Classification Techniques II

BUSINESS SCHOOL

QBUS6850 Team

THE UNIVERSITY OF SYDNEY

❑ Topics covered

- Bootstrap
- Ensemble classifier
- Bagging
- Boosting and adaboost
- Random forest
- Multi-class classification and its confusion matrix

References

❑ References

- Friedman et al., (2001), Chapters 9.2, 10, 15, 16
- James et al., (2014), Chapter 8
- Bishop, (2006), Chapters 14.3 - 14.4
- Alpaydin, (2014), Chapter 9

Learning Objectives

- ❑ Understand the intuition of bootstrap, ensemble classifier, Bagging, boosting and random forest work

- ❑ Understand how bootstrap, ensemble classifier, bagging, boosting and random forest work

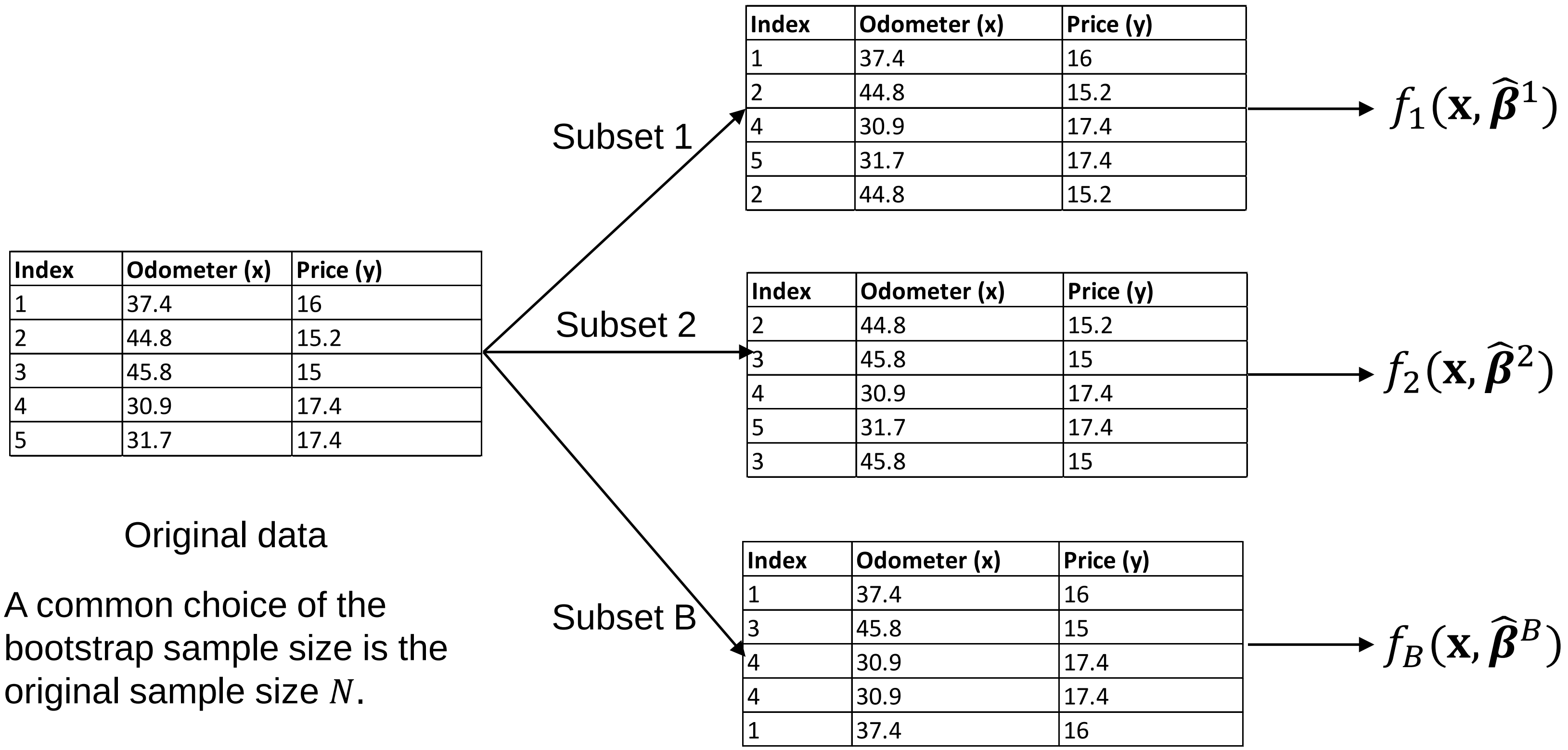
- ❑ Understand the one-vs-one and one-vs-all approach of multi-class classification

- ❑ Understand how to interpret the confusion matrix of multi-class classification problem

- ❑ Understand how to calculate the misclassification rate, accuracy, precision and recall of multi-class classification problem

Bootstrap

- The bootstrap is a widely applicable and extremely powerful statistical tool
- Bootstrap can be used to quantify the uncertainty associated with a given estimator or statistical learning method.
- Randomly select $N = 5$ observations (with replacement) from the original data set in order to produce a bootstrap data set



Index	Odometer (x)	Price (y)
1	37.4	16
2	44.8	15.2
3	45.8	15
4	30.9	17.4
5	31.7	17.4

Original data

Subset 1

Index	Odometer (x)	Price (y)
1	37.4	16
2	44.8	15.2
4	30.9	17.4
5	31.7	17.4
2	44.8	15.2

$f_1(\mathbf{x}, \widehat{\boldsymbol{\beta}}^1)$

Subset 2

Index	Odometer (x)	Price (y)
2	44.8	15.2
3	45.8	15
4	30.9	17.4
5	31.7	17.4
3	45.8	15

$f_2(\mathbf{x}, \widehat{\boldsymbol{\beta}}^2)$

Subset B

Index	Odometer (x)	Price (y)
1	37.4	16
3	45.8	15
4	30.9	17.4
4	30.9	17.4
1	37.4	16

$f_B(\mathbf{x}, \widehat{\boldsymbol{\beta}}^B)$

A common choice of the bootstrap sample size is the original sample size N.

Math Behind Bootstrap

Now with B different bootstrap data sets, we have B model estimates $f_1\left(\mathbf{x}, \widehat{\boldsymbol{\beta}}^1\right), f_2\left(\mathbf{x}, \widehat{\boldsymbol{\beta}}^2\right), \ldots, f_B\left(\mathbf{x}, \widehat{\boldsymbol{\beta}}^B\right)$

The mean of the B estimates $f_1\left(\mathbf{x}, \widehat{\boldsymbol{\beta}}^1\right), f_2\left(\mathbf{x}, \widehat{\boldsymbol{\beta}}^2\right), \ldots, f_B\left(\mathbf{x}, \widehat{\boldsymbol{\beta}}^B\right)$ is:

$$\bar{f}(\mathbf{x}, \boldsymbol{\beta}) = \frac{1}{B} \sum_{i=1}^{B} f_i\left(\mathbf{x}, \widehat{\boldsymbol{\beta}}^i\right)$$

The standard deviation of the B estimates is:

$$\sqrt{\frac{1}{B-1} \sum_{i=1}^{B} \left(f_i\left(\mathbf{x}, \widehat{\boldsymbol{\beta}}^i\right) - \bar{f}(\mathbf{x}, \boldsymbol{\beta})\right)^2}$$

The bootstrap approach can be used to **effectively** estimate the variability associated with $f(\mathbf{x}, \widehat{\boldsymbol{\beta}})$: good estimation of mean and SD compared to the true.

Ensemble Classifier

(Lecture07_Example01.py)

Ensemble Classifier

Power of the crowd

❑ Basic idea:

- Build different “experts”, and let them **vote**

❑ Advantages:

- Improve predictive performance
- Other types of classifiers can be directly included
- Easy to implement
- Not much parameter tuning

❑ Disadvantages:

- The combined classifier is not so transparent (black box)

Why does it work?

- ❑ Suppose there are 3 base classifiers
- ❑ Each classifier has misclassification rate: $\varepsilon = 25\%$
- ❑ Assume independence among classifiers
- ❑ Probability that the ensemble classifier makes a wrong prediction:
  - 2 or more than 2 classifiers make a wrong prediction

$$P(X \geq 2) = \sum_{i=2}^{3} \binom{3}{i} \varepsilon^{i} (1-\varepsilon)^{3-i} = 0.156$$

Considerably lower misclassification rate than the base classifiers
Usually, **the more classifiers the better**

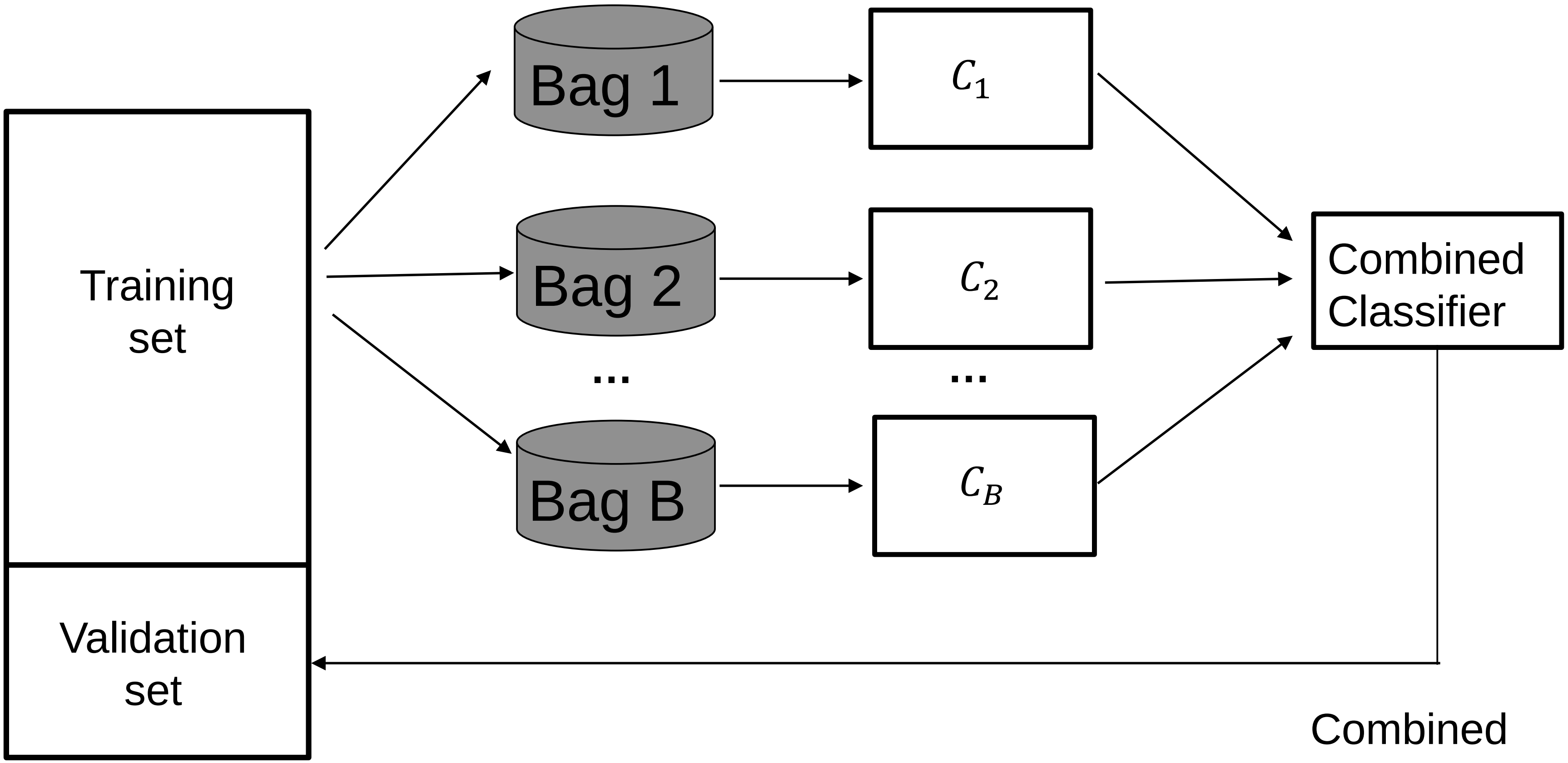
Bagging

Bagging

Voting/averaging a set of observations reduces variance

- ❑ **Bagging (Bootstrap aggregating)***:*
  - Sample **B bootstrap** samples, although we only have one dataset. This helps a lot if data is noisy.
  - Train **B different classifiers** (maybe they are weak classifiers, e.g high misclassification rate) on these bootstrap samples
  - For a new example, let all classifiers predict and take a majority vote (or average)
- ❑ The basic idea that this works is that if the classifiers make independent errors, then their ensemble can improve performance.
- ❑ The variance in the prediction is reduced (we don't suffer from the random errors that a single classifier is bound to make).

Intuition



Training set

Validation set

Bag 1

Bag 2

...

Bag B

C_1

C_2

...

C_B

Combined Classifier

Raw data

Multiple bootstrapped datasets

Multiple (maybe weak) classifiers

Combined (**strong**) classifier. Voting or averaging.

With bootstrap, by taking repeated samples from the (single) training data set, we generate B different bootstrapped training data sets.

We then train our method on the b-th bootstrapped training set in order to get $f_b(\mathbf{x}, \widehat{\boldsymbol{\beta}}^b)$, and finally average (or mode) all the predictions, to obtain

$$\bar{f}(\mathbf{x}, \boldsymbol{\beta}) = \frac{1}{B}\sum_{i=1}^{B} f_i(\mathbf{x}, \widehat{\boldsymbol{\beta}}^i) \qquad \bar{f}(\mathbf{x}, \boldsymbol{\beta}) = \mathrm{Mode}\{f_i(\mathbf{x}, \widehat{\boldsymbol{\beta}}^i)\}_{i=1}^{B}$$

Bagging can improve predictions for many regression/classification methods, it is **particularly useful for decision trees**.

Bagging has been demonstrated to give impressive improvements in accuracy by combining together hundreds or even thousands of trees into a single procedure.

When does Bagging work?

- ❑ Learning algorithm is suffering from high variance problem: if small changes to the training set cause large changes in the learned classifier.
- ❑ Bagging almost always improves performance
- ❑ Some candidates:
  - Decision tree
  - Regression tree
  - Linear regression
  - Logistic regression
  - SVM

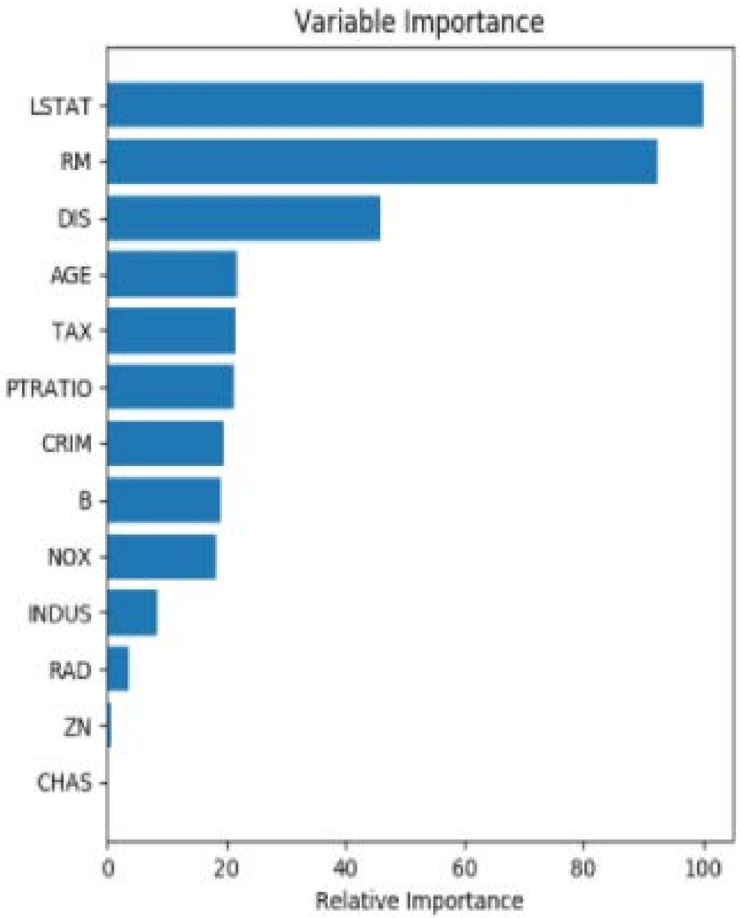
Disadvantages of Bagging

❑ Black box: it can be difficult to interpret the resulting model from bagging

❑ When we bag a large number of decision trees, it is no longer possible to represent the resulting statistical learning procedure using a single tree, and it is no longer clear which variables are most important to the procedure. Thus, bagging improves prediction accuracy at the expense of interpretability.

❑ How to overcome?

In the context of bagging classification trees, we can add up the total amount that the information gain by splits over a given feature, averaged over all B trees, to represent the importance of the feature



http://scikit-learn.org/stable/modules/ensemble.html

Boosting & Adaboost

Boosting

Power of the weighted crowd

- ❑ Similar as bagging, boosting is a general approach that can be applied to many statistical learning methods for regression or classification.

- ❑ Boosting works in a similar way, except that the trees are grown **sequentially** each tree is grown **using information from previously grown trees.**

- ❑ Boosting does not involve bootstrap sampling

- ❑ Each tree is fit on a **modified** version **(weighted)** of the original data set.

Boosting Intuition

❑ Train classifiers (e.g. decision trees) in a **sequence**

❑ Each classifier could be very “weak”, e.g. a decision stump

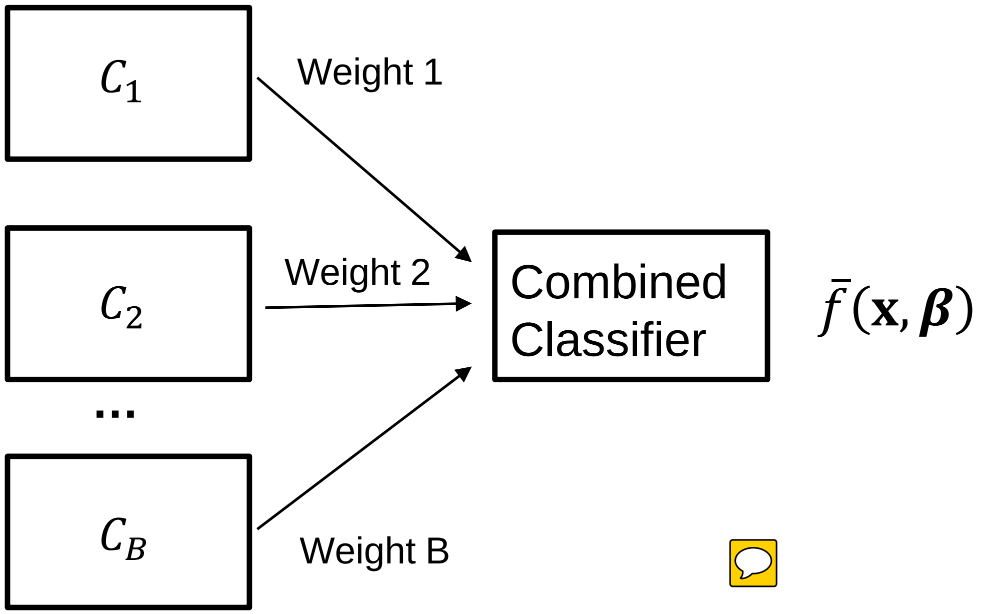
❑ A new classifier should **focus more (higher weights)** on those data points which were **incorrectly** classified in the last round. Data points which are wrongly classified get high weight (the algorithm will focus on them)

❑ Combine the classifiers by letting them vote on the final prediction

- These classifiers are weighed to combine them into a single powerful classifier.
- Classifiers that have low training misclassification/error rates have high weight

❑ Boosting is not limited to decision tress and can be used for many classifiers

Boosting Output



$$\bar{f}(\mathbf{x}, \boldsymbol{\beta}) = \operatorname{sign}\left(\boxed{\alpha_1} f_1\left(\mathbf{x}, \widehat{\boldsymbol{\beta}}^1\right) + \alpha_2 f_2\left(\mathbf{x}, \widehat{\boldsymbol{\beta}}^2\right) + \cdots + \alpha_B f_B\left(\mathbf{x}, \widehat{\boldsymbol{\beta}}^B\right)\right)$$

Weight or voting power of classifier 1

AdaBoost, short for "Adaptive Boosting", is a machine learning meta-algorithm formulated by Yoav Freund and Robert Schapire who won the Gödel Prize in 2003 for their work.

Boosting (Adaboost) Algorithm

- Training set: $$\mathcal{D} = \left\{ \begin{matrix} A(\mathbf{x}_1 = 1.6, t_1 = +1), B(\mathbf{x}_2 = 1.2, t_2 = +1), \\ C(\mathbf{x}_3 = 4.5, t_3 = -1), D(\mathbf{x}_4 = 7.2, t_4 = +1), \\ E(\mathbf{x}_5 = 6.3, t_5 = +1) \end{matrix} \right\}$$
- $t \in \{-1, +1\}$
- $N = 5, d = 1$. 5 training examples, 1 feature. Hence $\mathbf{x}_3 = x_{31}$ and the generic variable is $\mathbf{x} = x_1$

- 6 weak classifiers: decision stumps

$$f_1(\mathbf{x}, \widehat{\boldsymbol{\beta}}^1) = \begin{cases} +1 & if\ x_1 < 1 \\ -1 & if\ x_1 \geq 1 \end{cases} \qquad f_2(\mathbf{x}, \widehat{\boldsymbol{\beta}}^2) = \begin{cases} +1 & if\ x_1 < 2 \\ -1 & if\ x_1 \geq 2 \end{cases}$$

$$f_3(\mathbf{x}, \widehat{\boldsymbol{\beta}}^3) = \begin{cases} +1 & if\ x_1 < 6 \\ -1 & if\ x_1 \geq 6 \end{cases} \qquad f_4(\mathbf{x}, \widehat{\boldsymbol{\beta}}^4) = \begin{cases} +1 & if\ x_1 \geq 1 \\ -1 & if\ x_1 < 1 \end{cases}$$

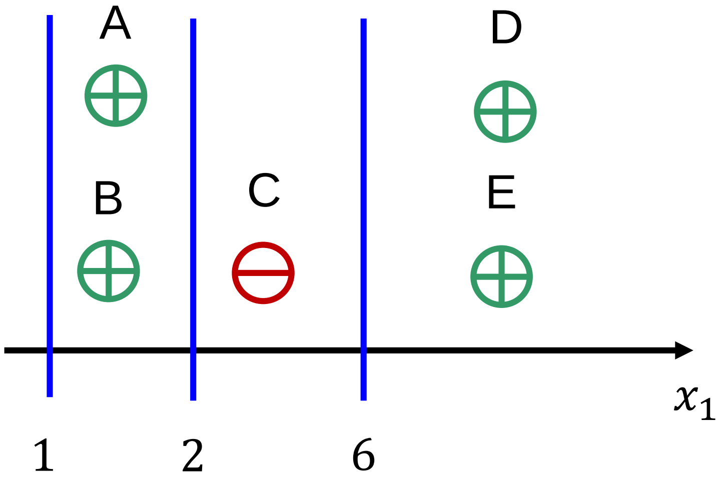
$$f_5(\mathbf{x}, \widehat{\boldsymbol{\beta}}^5) = \begin{cases} +1 & if\ x_1 \geq 2 \\ -1 & if\ x_1 < 2 \end{cases} \qquad f_6(\mathbf{x}, \widehat{\boldsymbol{\beta}}^6) = \begin{cases} +1 & if\ x_1 \geq 6 \\ -1 & if\ x_1 < 6 \end{cases}$$

Boosting (Adaboost) Algorithm

- If the first condition of decision stump is satisfied, the classifier predict +1 (positive), otherwise predict -1 (negative)
- Consider data $A(\mathbf{x}_1 = 1.6, t_1 = +1)$: We will have

$$f_1\left(\mathbf{x}_1, \widehat{\boldsymbol{\beta}}^1\right) = -1\ (\text{wrong});\ f_2\left(\mathbf{x}_1, \widehat{\boldsymbol{\beta}}^2\right) = +1;\ f_3\left(\mathbf{x}_1, \widehat{\boldsymbol{\beta}}^2\right) = +1$$

$$f_4\left(\mathbf{x}_1, \widehat{\boldsymbol{\beta}}^1\right) = +1;\ f_5\left(\mathbf{x}_1, \widehat{\boldsymbol{\beta}}^2\right) = -1\ (\text{wrong});\ f_6\left(\mathbf{x}_1, \widehat{\boldsymbol{\beta}}^2\right) = -1(\text{wrong})$$



Classifier	Misclassified points
$x_1 < 1$	A B D E
$x_1 < 2$	D E
$x_1 < 6$	C D E
$x_1 \geq 1$	C
$x_1 \geq 2$	A B C
$x_1 \geq 6$	A B

Algorithm- iteration 1

1.1 Initialize weights of training examples. Equal weights as no prior information.

$$w_i = \frac{1}{N}$$

1.2 Calculate misclassification rate ε_i for each classifier $f_i(\mathbf{x}, \widehat{\boldsymbol{\beta}}^i)$. 6 classifiers in our example.

1.3 Pick the $f_i(\mathbf{x}, \widehat{\boldsymbol{\beta}}^i)$ with the lowest misclassification rate: $f_4(\mathbf{x}, \widehat{\boldsymbol{\beta}}^4)$.

Training examples	Weights iter 1
A	1/5
B	1/5
C	1/5
D	1/5
E	1/5

Classifier	Misclassified points	Misclassification rate
$x_1 < 1$	A B D E	4/5
$x_1 < 2$	D E	2/5
$x_1 < 6$	C D E	3/5
$x_1 \geq 1$	**C**	**1/5**
$x_1 \geq 2$	A B C	3/5
$x_1 \geq 6$	A B	2/5

Algorithm- iteration 1

1.4 Calculate voting power for the best classifier $f_i(\mathbf{x}, \widehat{\boldsymbol{\beta}}^i)$ $i = 4$. Lower misclassification rate, higher voting power. Natural log.

$$\alpha_i = \frac{1}{2}\log\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right), \qquad i = 4$$

1.5 Check whether the stopping criteria are met.

- ❑ If yes, stop.
  - Combined classifier $\bar{f}(\mathbf{x}, \boldsymbol{\beta})$ is good enough
  - Enough # of iterations
  - No good classifier left, e.g. best $f_i(\mathbf{x}, \widehat{\boldsymbol{\beta}}^i)$ has misclassification rate 0.5
- ❑ If no, update the weights to examples that are misclassified. **$\boldsymbol{\varepsilon}$ is the misclassification rate of the best classifier**.

$$w_{new} = \begin{cases} \dfrac{1}{2(1-\varepsilon)} w_{old}, & \text{if} \quad \text{correct} \\ \dfrac{1}{2\varepsilon} w_{old}, & \text{if} \quad \text{incorrect} \end{cases}$$

Algorithm- iteration 1

For example, the new weight of C (incorrectly classified by model 4)

$$\frac{1}{2\varepsilon} w_{old} = \frac{1}{2(1/5)} \frac{1}{5} = \frac{1}{2}$$

$$\sum_{correct} w = \sum_{incorrect} w = \frac{1}{2}$$

For example, the new weight of A (correctly classified by model 4): Same for B, D, E

$$\frac{1}{2(1-\varepsilon)} w_{old} = \frac{1}{2(1-1/5)} \frac{1}{5} = \frac{1}{8}$$

Training examples	Weights iter 1	Weights iter 2
A	1/5	1/8
B	1/5	1/8
C	**1/5**	**1/2**
D	1/5	1/8
E	1/5	1/8

Classifier	Misclassified points	Misclassification rate	Voting power
$x_1 < 1$	A B D E	4/5	
$x_1 < 2$	D E	2/5	
$x_1 < 6$	C D E	3/5	
$x_1 \geq 1$	**C**	**1/5**	**1/2log(4)**
$x_1 \geq 2$	A B C	3/5	
$x_1 \geq 6$	A B	2/5	

C is the only point that is misclassified in iteration 1, which receives higher weight in the next iteration.

Current classifier: $\frac{1}{2}\log(4)\ f_4\left(\mathbf{x}, \widehat{\boldsymbol{\beta}}^4\right)$

Algorithm- iteration 2

The classifier trained in iteration 1 is not good enough. Then go to iteration 2.

2.2 Again, calculate misclassification rate ε_i for each classifier $f_i(\mathbf{x}, \widehat{\boldsymbol{\beta}}^i)$, $i = 1, \dots, 6$, by imply taking the sum of weights of misclassified data.

For example, for classifier 3, the sum of weights of C, D,E is 4/8+1/8+1/8=6/8

And Classifiers used in last iteration will have $\varepsilon_i = 1/2$

2.3 Pick the $f_i(\mathbf{x}, \widehat{\boldsymbol{\beta}}^i)$with the lowest misclassification rate. If have a draw, pick up the first one: $f_2(\mathbf{x}, \widehat{\boldsymbol{\beta}}^2)$.

Training examples	Weights iter 2
A	1/8
B	1/8
C	**½=4/8**
D	1/8
E	1/8

Classifier	Misclassified points	Misclassification rate- iter 2	Voting power
$x_1 < 1$	A B D E	(1/8)*4=1/2	
$x_1 < 2$	**D E**	**2/8**	
$x_1 < 6$	C D E	6/8	
$x_1 \geq 1$	C	4/8=1/2	1/2log(4)
$x_1 \geq 2$	A B C	6/8	
$x_1 \geq 6$	**A B**	**2/8**	

Algorithm- iteration 2

2.4 Calculate voting power for the best classifier $f_2(\mathbf{x}, \widehat{\boldsymbol{\beta}}^2)$.

$$\alpha_2 = \frac{1}{2}\log\left(\frac{1-\varepsilon_2}{\varepsilon_2}\right) = \frac{1}{2}\log\left(\frac{1-2/8}{2/8}\right) = \frac{1}{2}\log(3)$$

Training examples	Weights iter 2
A	1/8
B	1/8
C	½=4/8
D	**1/8**
E	**1/8**

Classifier	Misclassified points	Misclassification rate- iter 2	Voting power
$x_1 < 1$	A B D E	(1/8)*4=1/2	
$x_1 < 2$	D E	2/8	1/2log(3)
$x_1 < 6$	C D E	6/8	
$x_1 \geq 1$	C	4/8=1/2	1/2log(4)
$x_1 \geq 2$	A B C	6/8	
$x_1 \geq 6$	**A B**	**2/8**	

D and E will receive higher weight in the next iteration.

Current classifier: $\frac{1}{2}\log(4)\ f_4(\mathbf{x}, \widehat{\boldsymbol{\beta}}^4) + \frac{1}{2}\log(3)\ f_2(\mathbf{x}, \widehat{\boldsymbol{\beta}}^2)$

Algorithm- iteration 2

2.5 Check whether the stopping criteria are met.

- If no, update the weights to examples that are misclassified. ε is the misclassification rate of the best classifier.

$$w_{new} = \begin{cases} \frac{1}{2(1-\varepsilon)} w_{old}, & \text{if} \quad \text{correct} \\ \frac{1}{2\varepsilon} w_{old}, & \text{if} \quad \text{incorrect} \end{cases}$$

For example, the new weight of C (correctly classified by classifier 2)

$$\frac{1}{2(1-\varepsilon)} w_{old} = \frac{1}{2(1-2/8)} \frac{4}{8} = \frac{4}{12}$$

Training examples	Weights iter 1	Weights iter 2	Weights iter 3
A	1/5	1/8	1/12
B	1/5	1/8	1/12
C	1/5	½=4/8	4/12
D	**1/5**	**1/8**	**¼=3/12**
E	**1/5**	**1/8**	**¼=3/12**

$$\sum_{correct} w = \sum_{incorrect} w = \frac{1}{2}$$

So here we can get the new weight for D and E quickly with the above formula

Algorithm- iteration 3

The classifier trained in iteration 2 is not good enough. Then go to iteration 3.

3.2 Again, calculate misclassification rate ε_i for each classifier $f_i(\mathbf{x}, \widehat{\boldsymbol{\beta}}^i)$. Classifiers used in last iteration will have $\varepsilon_i = 1/2$

3.3 Pick the $f_i(\mathbf{x}, \widehat{\boldsymbol{\beta}}^i)$ with the lowest misclassification rate: $f_6(\mathbf{x}, \widehat{\boldsymbol{\beta}}^6)$.

Training examples	Weights iter 3
A	1/12
B	1/12
C	4/12
D	**¼=3/12**
E	**¼=3/12**

Classifier	Misclassified points	Misclassification rate- iter 3	Voting power
$x_1 < 1$	A B D E	1/12+1/12+3/12+3/12= 8/12	
$x_1 < 2$	D E	6/12	1/2log(3)
$x_1 < 6$	C D E	10/12	
$x_1 \geq 1$	C	4/12	1/2log(4)
$x_1 \geq 2$	A B C	6/12	
$x_1 \geq 6$	**A B**	**2/12**	

Algorithm- iteration 3

2.4 Calculate voting power for the best classifier $f_6(\mathbf{x}, \widehat{\boldsymbol{\beta}}^6)$. $\alpha_i = \frac{1}{2}\log\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$

Training examples	Weights iter 3
A	1/12
B	1/12
C	4/12
D	**¼=3/12**
E	**¼=3/12**

Classifier	Misclassified points	Misclassification rate- iter 3	Voting power
$x_1 < 1$	A B D E	1/12+1/12+3/12+3/12= 8/12	
$x_1 < 2$	D E	6/12	1/2log(3)
$x_1 < 6$	C D E	10/12	
$x_1 \geq 1$	C	4/12	1/2log(4)
$x_1 \geq 2$	A B C	6/12	
$x_1 \geq 6$	**A B**	**2/12**	1/2log(5)

Best classifier: $\text{sign}\left(\frac{1}{2}\log(4)\ f_4\left(\mathbf{x}, \widehat{\boldsymbol{\beta}}^4\right) + \frac{1}{2}\log(3)\ f_2\left(\mathbf{x}, \widehat{\boldsymbol{\beta}}^2\right) + \frac{1}{2}\log(5)\ f_6\left(\mathbf{x}, \widehat{\boldsymbol{\beta}}^6\right)\right)$

Algorithm- iteration 3

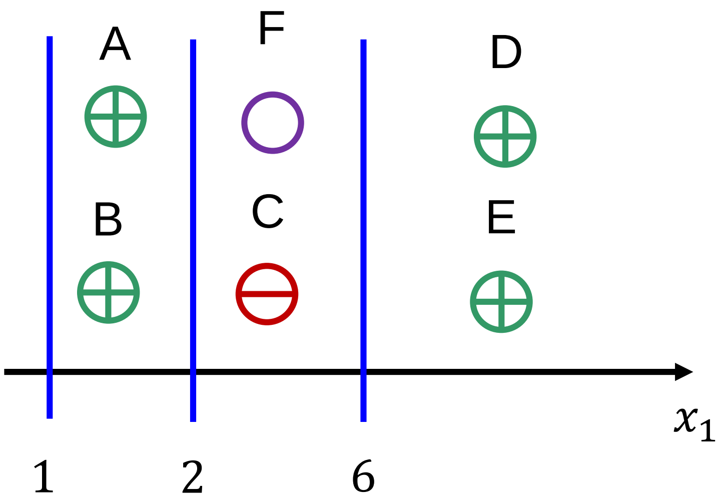
- **Best classifier:** $\operatorname{sign}\left(\frac{1}{2}\log(4)\ f_4\left(\mathbf{x},\widehat{\boldsymbol{\beta}}^4\right)+\frac{1}{2}\log(3)\ f_2\left(\mathbf{x},\widehat{\boldsymbol{\beta}}^2\right)+\frac{1}{2}\log(5)\ f_6\left(\mathbf{x},\widehat{\boldsymbol{\beta}}^6\right)\right)$
- It can classify all training examples correctly.
- Why?
- For a new test example F($\mathbf{x}^* = 3.2$), class prediction?

$$\operatorname{sign}\left(\frac{1}{2}\log(4)\ f_4\left(\mathbf{x}^*,\widehat{\boldsymbol{\beta}}^4\right)+\frac{1}{2}\log(3)\ f_2\left(\mathbf{x}^*,\widehat{\boldsymbol{\beta}}^2\right)+\frac{1}{2}\log(5)\ f_6\left(\mathbf{x}^*,\widehat{\boldsymbol{\beta}}^6\right)\right)$$

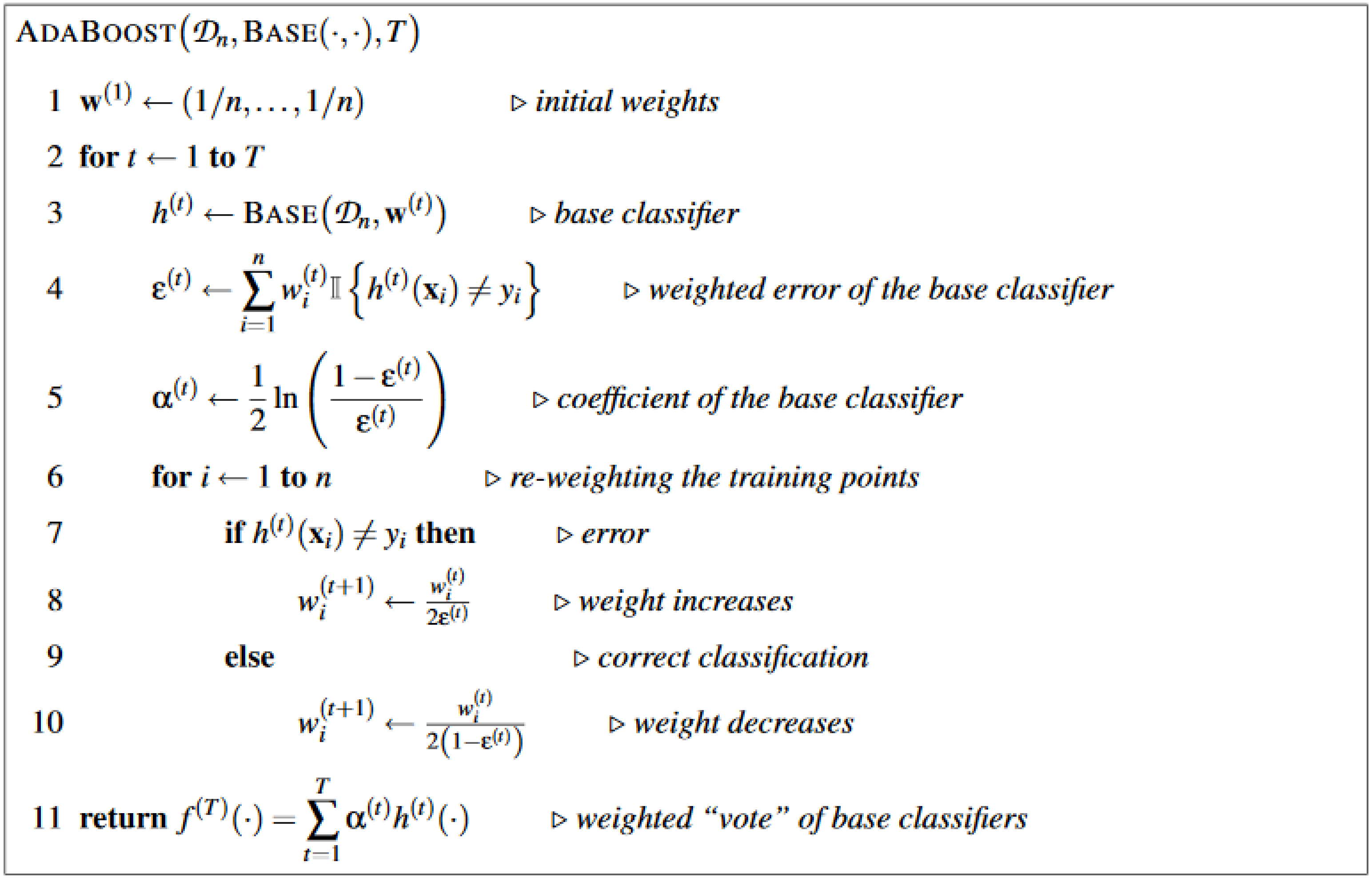
$$= \operatorname{sign}\left(\frac{1}{2}\left(\log(4)\ (+1)+\log(3)\ (-1)+\log(5)\ (-1)\right)\right) = sign(-0.6609) = -1.$$

Predicting F negative

Classifier	Misclassified points
$x_1 < 1$	A B D E
$x_1 < 2$	**D E**
$x_1 < 6$	C D E
$x_1 \geq 1$	**C**
$x_1 \geq 2$	A B C
$x_1 \geq 6$	**A B**



Algorithm Summary



```
ADABOOST($\mathcal{D}_n$, BASE$(\cdot,\cdot)$, $T$)

$\mathbf{w}^{(1)} \leftarrow (1/n,\ldots,1/n)$        ▷ initial weights
for $t \leftarrow 1$ to $T$
    $h^{(t)} \leftarrow$ BASE$(\mathcal{D}_n, \mathbf{w}^{(t)})$        ▷ base classifier
    $\varepsilon^{(t)} \leftarrow \sum_{i=1}^{n} w_i^{(t)} \mathbb{I}\left\{h^{(t)}(\mathbf{x}_i) \neq y_i\right\}$        ▷ weighted error of the base classifier
    $\alpha^{(t)} \leftarrow \frac{1}{2}\ln\left(\frac{1-\varepsilon^{(t)}}{\varepsilon^{(t)}}\right)$        ▷ coefficient of the base classifier
    for $i \leftarrow 1$ to $n$        ▷ re-weighting the training points
        if $h^{(t)}(\mathbf{x}_i) \neq y_i$ then        ▷ error
            $w_i^{(t+1)} \leftarrow \frac{w_i^{(t)}}{2\varepsilon^{(t)}}$        ▷ weight increases
        else        ▷ correct classification
            $w_i^{(t+1)} \leftarrow \frac{w_i^{(t)}}{2(1-\varepsilon^{(t)})}$        ▷ weight decreases
return $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$        ▷ weighted “vote” of base classifiers
```

From https://users.lal.in2p3.fr/kegl/teaching/stages/notes/tutorial

Random Forest

THE UNIVERSITY OF SYDNEY

Random Forest

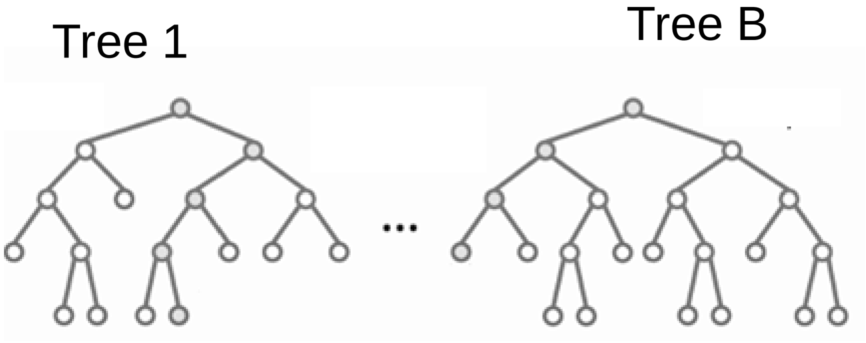
❑**Random forest** (or **random forests**) is an ensemble classifier that consists of many decision trees and outputs the class that is the mode of the class's output by individual trees.

❑The term came from random decision forests that was first proposed by Tin Kam Ho of Bell Labs in 1995.

❑The method combines Breiman's "**bagging**" idea and the **random selection of features.**

❑Random forests provide an improvement over bagged trees by way of a random small tweak that **decorrelates** the trees.

Random Forests



- Random forests (RF) are a combination of tree predictors
- Each tree depends on the values of a random set sampled in dependently
- The generalization error depends on the strength of the individual trees and the correlation between them

Random Forest

❑Random forests provide an improvement over bagged trees by way of a random small tweak that decorrelates the trees

❑In bagging, we build a number forest of decision trees on bootstrapped training samples

❑Each time a split in a tree is considered, a random sample of $\boldsymbol{p}$ features is chosen as split candidates from the full set of $\boldsymbol{d}$ features

❑**In RF, the number of features considered at each split is approximately equal to the square root of the total number of features**

❑To avoid the situation that in bagging there is a quite strong feature, resulting most or all of the trees will use this strong predictor in the top split and produce very similar trees

❑Random forests overcome this problem by forcing each split to consider only a subset of the features

Algorithm

1. For tree $b = 1$ to B:

   (a) Choose a bootstrap sample of size N from training data

   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each leaf node of the tree, until the minimum node size is achieved:

      i. Select p variables at random from the d variables ($p \leq d$).**Why doing this?**

      ii. Pick the best variable/split-point among the p.

      iii. Split the node into two decision nodes.

2. Output the ensemble of trees $\{T_b\}_1^B$.

Friedman et al., (2001)

RF Prediction

To make a prediction at a new point $\mathbf{x}$:

For regression:

$$\bar{f}(\mathbf{x}, \boldsymbol{\beta}) = \frac{1}{B}\sum_{b=1}^{B} T_b(\mathbf{x})$$

For classification:
Suppose the class prediction of the b_{th} random-forest tree is $C_b(\mathbf{x})$:

$$\bar{f}(\mathbf{x}, \boldsymbol{\beta}) = \ \mathrm{Mode}\{C_b(\mathbf{x})\}_{b=1}^{B}$$

Why RF works?

- Decision trees are ideal candidates for bagging, since they can capture complex interaction structures in the data, and if grown sufficiently deep, have relatively low bias.
- Since each tree generated in bagging is identically distributed (i.d.), the expectation of an average of B such trees is the same as the expectation of any one of them.
- This means the bias of bagged trees is the same as that of the individual trees, and the only hope of improvement is through variance reduction.
- This is in contrast to boosting, where the trees are grown in an adaptive way to remove bias, and hence are not i.d.

Why RF works?

An average of B i.i.d. random variables, each with variance σ^2, has variance:

$$\frac{1}{B}\sigma^2$$

If the variables are simply i.d. (identically distributed, but not necessarily independent) with positive pairwise correlation ρ, the variance of the average is

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

As B increases, the second term disappears, but the first remains, and hence the size of the **correlation** of pairs of bagged trees limits the benefits of averaging.

Why RF works?

- The idea in random forests is to improve the variance reduction of bagging by reducing the correlation between the trees, without increasing the variance too much.

- This is achieved in the tree-growing process through random selection of the input variables.

Select p variables at random from the d variables ($p \leq d$)

- Typically values for p are $\sqrt{d}$ or even as low as 1.

- Intuitively, reducing p will reduce the correlation between any pair of trees in the ensemble, and hence the RF algorithm **reduces the variance** of the average.

Recommendations

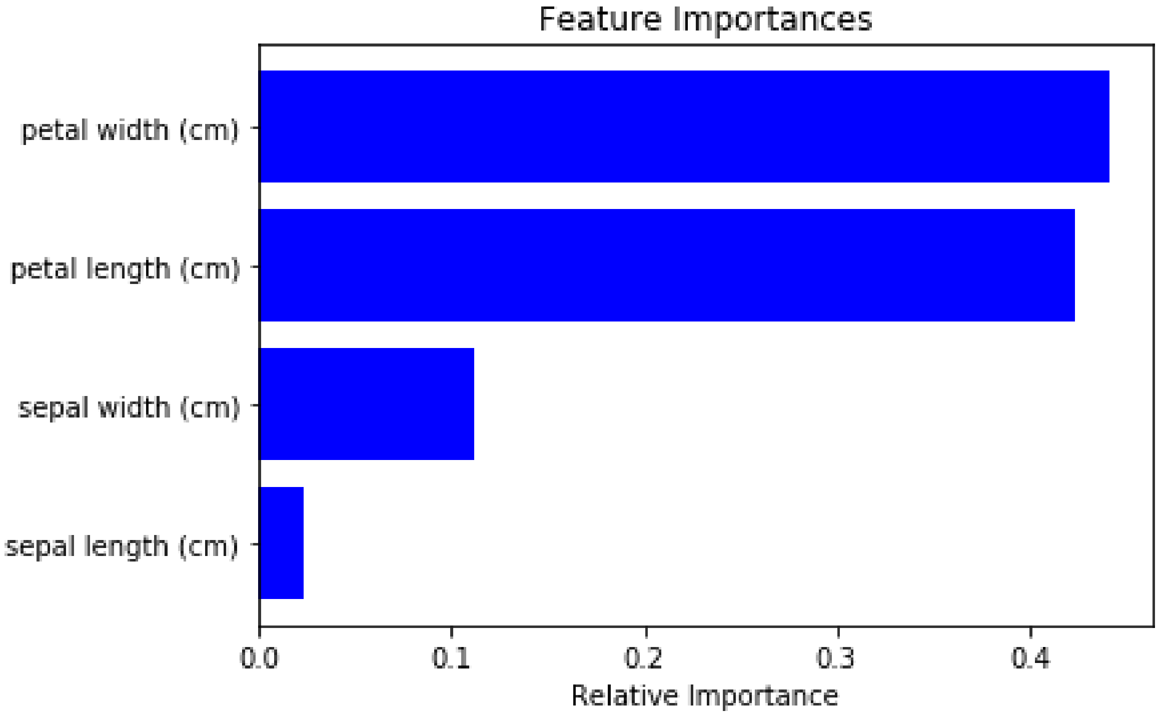
- For classification, the default value for p is $\sqrt{d}$ and the minimum node size is one.

- For regression, the default value for p is $d/3$ and the minimum node size is five.

Friedman et al., (2001)

Feature Importance

At each split in each tree, the improvement in the split-criterion is the importance measure attributed to the splitting variable, and is accumulated over all the trees in the forest separately for each variable. *Lecture07_Example02.py*



RF Compared with Boosting

□ Advantages:

- It is more robust.
- It is faster to train (no reweighting, each split is on a small subset of data and feature).
- Can handle missing/partial data.
- Is easier to extend to online version.

□ Disadvantages:

- The feature selection process is not explicit.
- Has weaker performance on small size training data.

Features and Advantages

The advantages of random forest are:

- It is one of the most accurate learning algorithms available. For many data sets, it produces a highly accurate classifier.
- It runs efficiently on large databases.
- It can handle thousands of input variables without variable deletion.
- It gives estimates of what variables are important in the classification.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.

https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#overview

Conclusions & Summary:

- ❑ Fully **parallelizable**
- ❑ Automatic predictor selection from large number of candidates
- ❑ Resistance to overfitting
- ❑ Ability to handle data without preprocessing
  - data does not need to be rescaled, transformed, or modified
  - resistant to outliers
  - automatic handling of missing values

Multi-class Classification

Multi-class Classification

❑ **Multi-class classification**

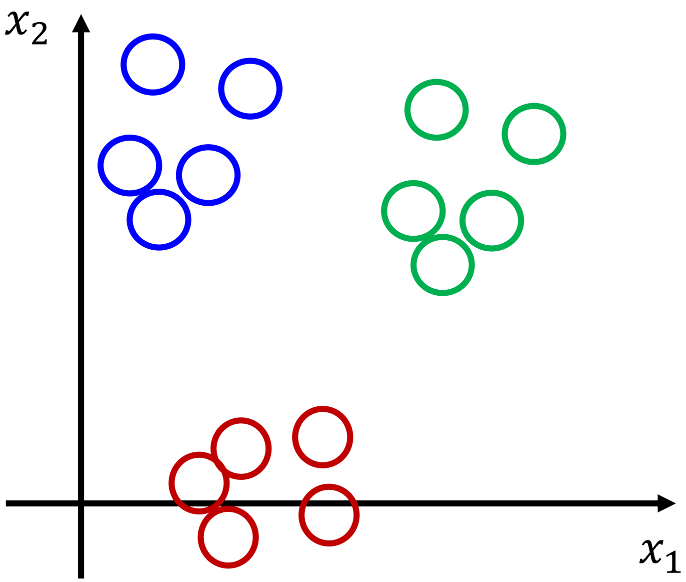
- In machine learning, multiclass or multinomial classification is the problem of classifying instances into one of the more than two classes (classifying instances into one of the two classes is called binary classification).

❑ **One-vs-one approach**

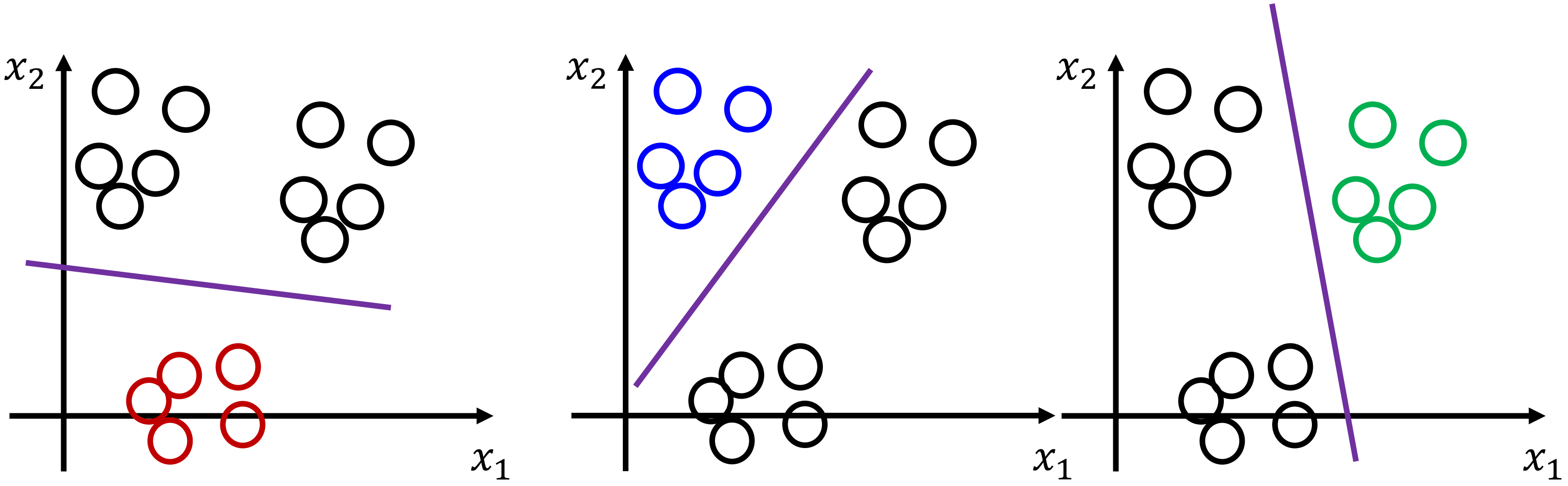
- Classifier is built for every pairwise combination of classes. This gives us a total of $K(K-1)/2$ binary classifiers. K is the number of classes.
- Each classifier receives the samples of a pair of classes from the original training set, and learn to distinguish these two classes. At prediction time, a **voting** scheme is applied: all $K(K-1)/2$ classifiers are applied to an new example and the class that got the highest number of "$+1$" predictions gets predicted by the combined classifier

One-vs-all Approach

- **One-vs-all (one-vs-rest) approach**
  - Classifier is built for each class and the remaining classes are grouped into one large "negative" class.



- A binary classifier is built $f_k(\mathbf{x}, \widehat{\boldsymbol{\beta}}^k)$ for each class k to predict the class (or probability for logistic regression) of t belongs to class k
- For a new example $\mathbf{x}$, predicting the label k for which the corresponding classifier has the highest confidence score



As a Regression Problem

- ❑ Logistic Regression modelling converts a binary classification problem into a regression where the regression function is the probability (or score) for being a positive case
- ❑ The loss function is defined as the "distance" between two distributions. For a single case, this loss is

Don't forget this negative sign

$$-\left[t_n \log\left(f(\mathbf{x}_n, \boldsymbol{\beta})\right) + (1 - t_n) \log\left(1 - f(\mathbf{x}_n, \boldsymbol{\beta})\right)\right]$$

where $f(\mathbf{x}_n, \boldsymbol{\beta})$, defined as logistic function, is the probability for $\mathbf{x}_n$ to be "positive" case $t_n (= 1)$.

- ❑ Define

$$P_1(\mathbf{x}_n, \boldsymbol{\beta}) = f(\mathbf{x}_n, \boldsymbol{\beta}) \text{ and } P_2(\mathbf{x}_n, \beta) = 1 - f(\mathbf{x}_n, \beta): \; (P_1(\mathbf{x}_n, \boldsymbol{\beta}) + P_2(\mathbf{x}_n, \boldsymbol{\beta}) = 1)$$

$$\mathbf{t}_n = (t_{n1}, t_{n2}) \text{ where } t_{n1} = t_n; t_{n2} = 1 - t_n \; (i.e. \; t_{n1} + t_{n2} = 1)$$

- ❑ Two discrete distributions (two classes) $P_n = (P_1, P_2)$ and $\mathbf{t}_n$. Their 'distance' is defined as (the cross entropy)

$$-\left[t_{n1} \log\left(P_1(\mathbf{x}_n, \boldsymbol{\beta})\right) + t_{n2} \log\left(P_2(\mathbf{x}_n, \boldsymbol{\beta})\right)\right]$$

As a Regression Problem

❑ Overall Loss Function

$$L(\boldsymbol{\beta}) = -\sum_{n=1}^{N}\left[t_{n1}\log\big(P_1(\mathbf{x}_n,\boldsymbol{\beta})\big) + t_{n2}\log\big(P_2(\mathbf{x}_n,\boldsymbol{\beta})\big)\right]$$

❑ If we have K classes, then we shall build a discrete distribution with K components satisfying $P_k(\mathbf{x}_n,\boldsymbol{\beta}) \geq 0$ and $\sum_{k=1}^{K} P_k(\mathbf{x}_n,\boldsymbol{\beta}) \equiv 1$

❑ How can we achieve this? We will build K models such that

$$P_k(\mathbf{x}_n,\boldsymbol{\beta}) = \frac{e^{f_k(\mathbf{x}_n,\boldsymbol{\beta})}}{\sum_{j=1}^{K} e^{f_k(\mathbf{x}_n,\boldsymbol{\beta})}}$$

❑ The overall loss function

$$L(\boldsymbol{\beta}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\log\big(P_k(\mathbf{x}_n,\boldsymbol{\beta})\big)$$

where $\mathbf{t}_\mathrm{n} = (\mathrm{t}_{\mathrm{n}1},\mathrm{t}_{\mathrm{n}2},\dots,\mathrm{t}_{\mathrm{nK}})^\mathrm{T}$ is the hot-one vector for the label of case n

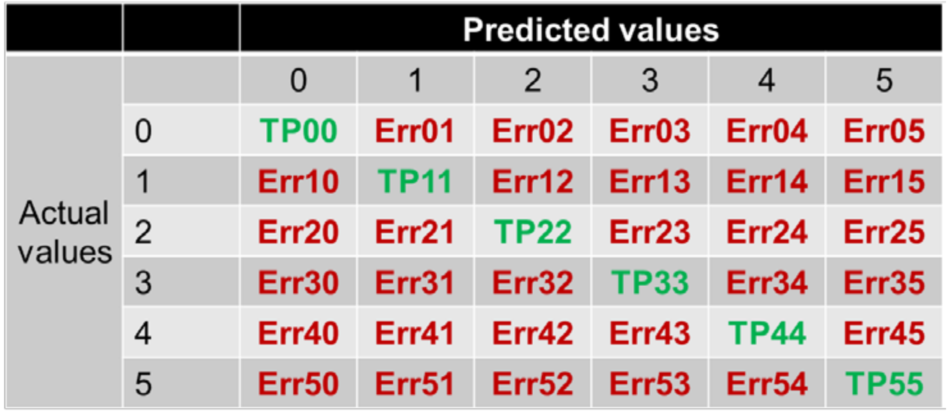
Confusion-matrix

Multi-class classification confusion-matrix

		Predicted values					
Actual values		0	1	2	3	4	5
	0	TP00	Err01	Err02	Err03	Err04	Err05
	1	Err10	TP11	Err12	Err13	Err14	Err15
	2	Err20	Err21	TP22	Err23	Err24	Err25
	3	Err30	Err31	Err32	TP33	Err34	Err35
	4	Err40	Err41	Err42	Err43	TP44	Err45
	5	Err50	Err51	Err52	Err53	Err54	TP55

For each class (treat it as positive class and the rest classes as negative),

- Sum of corresponding column is the total number of predictions;
- Sum of corresponding row is the total number of actual examples;
- Excluding the true positives, sum of **corresponding row** is the total number of **False Negatives (FN)**
- Excluding the true positives, sum of **corresponding column** is the total number of **False Positives (FP)**
- Excluding the class's column and row, sum of all columns and rows is the total number of True Negatives (TN)



		Predicted values					
		0	1	2	3	4	5
Actual values	0	TP00	Err01	Err02	Err03	Err04	Err05
	1	Err10	TP11	Err12	Err13	Err14	Err15
	2	Err20	Err21	TP22	Err23	Err24	Err25
	3	Err30	Err31	Err32	TP33	Err34	Err35
	4	Err40	Err41	Err42	Err43	TP44	Err45
	5	Err50	Err51	Err52	Err53	Err54	TP55

Measurements

- Accuracy:

$$\frac{\text{Total \#of correct classifications}}{\text{Total \# of data}}$$

- Misclassification (error) rate:

$$\frac{\text{Total \#of wrong classifications}}{\text{Total \# of data}}$$

- Precision

$$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$$

- Recall

$$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$$

Example

Treating class 0 as the positive class and the rest classes as negative:

Precision 0 =

$$\frac{\mathrm{TP00}}{\boxed{\mathrm{TP00+Err10+Err20++Err30+Err40+Err50}}}$$

Column sum

Recall 0=

$$\frac{\mathrm{TP00}}{\boxed{\mathrm{TP00+Err01+Err02++Err03+Err04+Err05}}}$$

Row sum

		Predicted values					
		0	1	2	3	4	5
Actual values	0	TP00	Err01	Err02	Err03	Err04	Err05
	1	Err10	TP11	Err12	Err13	Err14	Err15
	2	Err20	Err21	TP22	Err23	Err24	Err25
	3	Err30	Err31	Err32	TP33	Err34	Err35
	4	Err40	Err41	Err42	Err43	TP44	Err45
	5	Err50	Err51	Err52	Err53	Err54	TP55

Example

		Predicted values		
Actual values		0	1	2
	0	**15**	**6**	**2**
	1	**1**	**25**	**3**
	2	**5**	**0**	**29**

- Accuracy:

$$\frac{\text{Total \#of correct classifications}}{\text{Total \# of classification}}$$

$$= \frac{15 + 25 + 29}{86} = 80.2\%$$

- Misclassification rate=$1 - 80.2\%$=19.8%

- Precision 0

$$\frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{15}{15 + 1 + 5}$$

- Recall 2

$$\frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{29}{5 + 0 + 29}$$